Automated Machine Learning

Frank Zijun Zhang, PhD
Division of AI in medicine, Cedars-Sinai Medical Center

UCLA QCBio Collaboratory
AutoML workshop, Winter 2023

Workshop Overview

- This is a two-day *applied* course for building *convolutional neural networks* and *AutoML* in *genomics*.

  - applied: focus on implementations and coding, with a high-level theoretical understanding

  - CNNs: deep learning models that are good at learning spatial patterns (spatial inductive bias)

  - AutoML: automate the process of deep learning design

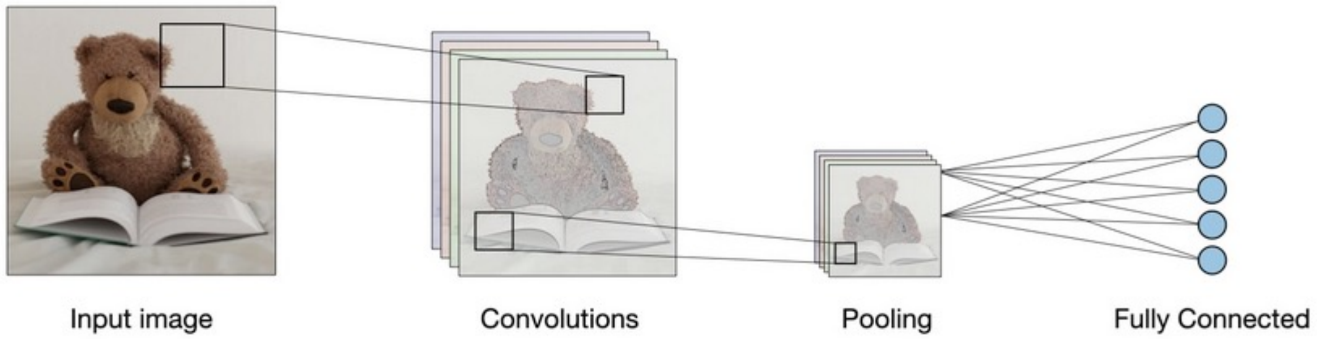
  - genomics: input DNA sequences, output biological insights

Workshop Goals

- Understand how CNNs model genomic sequences
- Understand the basic concepts in AutoML
- Implement CNNs in PyTorch and Tensorflow
- Implement AutoML prototype in AMBER
- Implement basic model explanation techniques

Agenda

- Day1: Build CNNs and Model Evaluation
  - Understand how CNNs model genomic sequences
  - Build CNNs with Tensorflow and PyTorch
  - Model evaluations and tuning

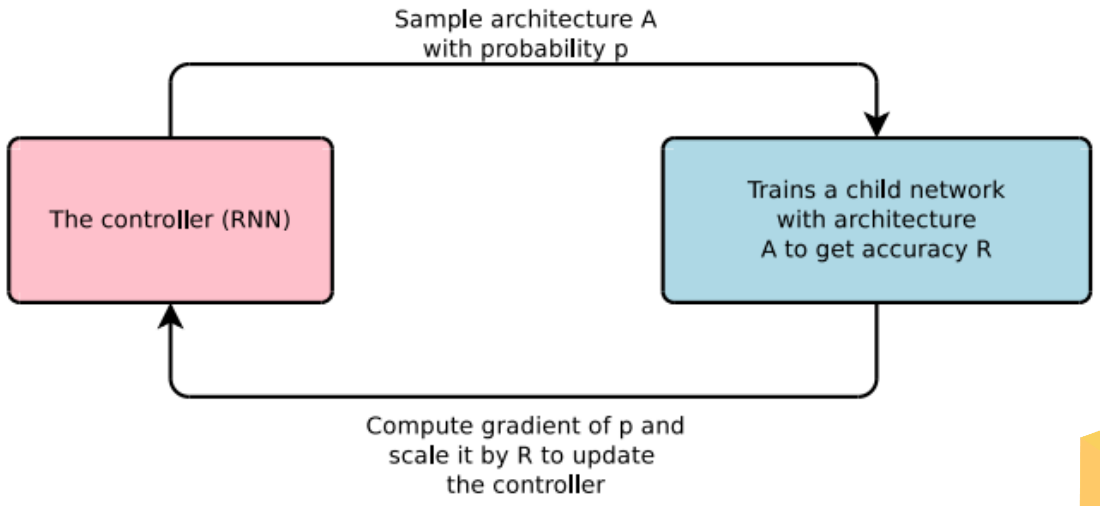


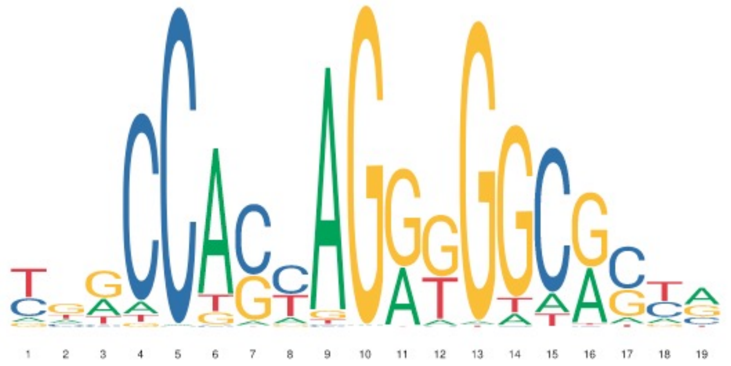
TensorFlow Keras PyTorch Lightning AI

https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks

Agenda

- Day2: AutoML and Model Explanation
  - Introduction of reinforcement learning
  - Apply NAS to automate CNN tuning
  - Interpret model and sequence motifs


Sample architecture A with probability p
The controller (RNN)
Trains a child network with architecture A to get accuracy R
Compute gradient of p and scale it by R to update the controller



AMBER

Zoph and Le, ICLR, 2017

A note on Google Colab and Github

- Go to Github Repo:
- https://github.com/zhanglab-aim/ucla-automl-workshop

- Open “UCLA_AutoML_workshop_Day1.ipynb”.

- Click on the Icon Open in Colab

- Each practice is ~10min; basic code is already implemented in the Notebook

- Explore the Practice questions shown in the slides.

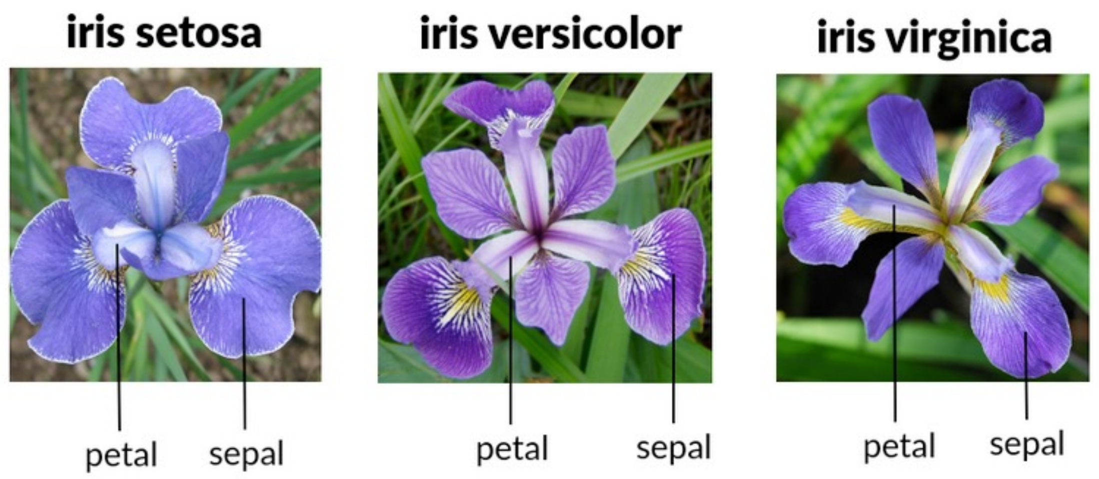
Day 1: Build CNNs and Model Evaluation

Frank Zijun Zhang, PhD
Division of AI in medicine, Cedars-Sinai Medical Center

UCLA QCBio Collaboratory
AutoML workshop, Winter 2023

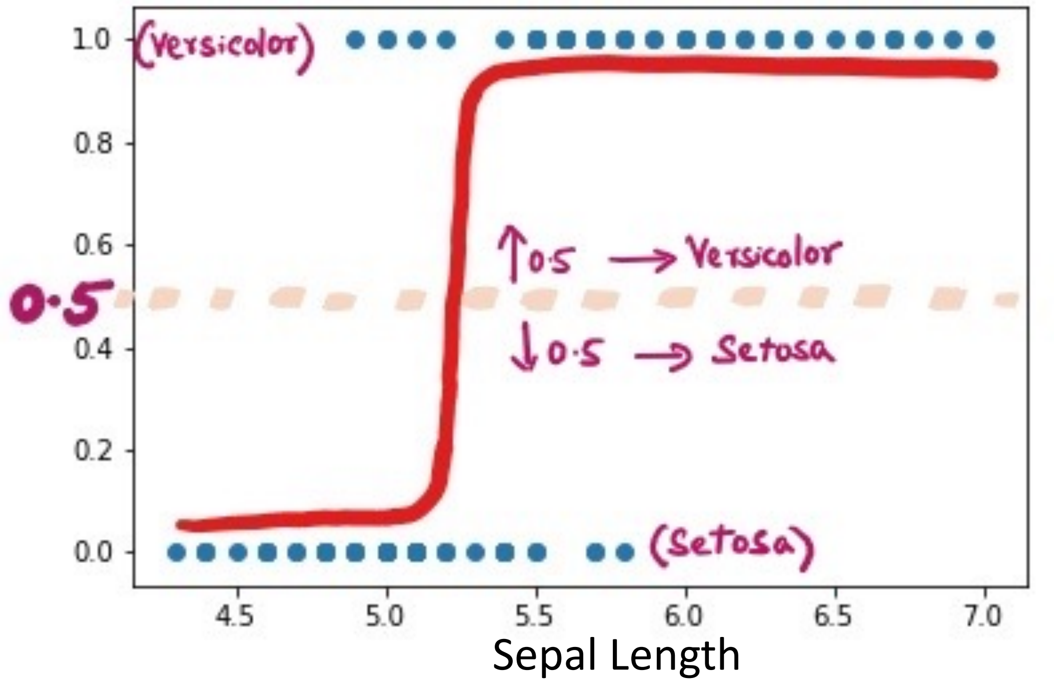
Recap: machine learning

- ML gives computers the ability to learn without being explicitly programmed.. *but how?*
- Machine learning originated from statistical learning.



$$P(Y = 1) = sigmoid\ (\beta_0 + \beta_1 \times petal + \beta_2 \times sepal)$$

betas are optimized by maximum likelihood.



https://ajaytech.co/2020/02/20/logistic-regression-2/

Recap: machine learning

- Learning Betas with gradient descent

- Optimization: maximize the likelihood on the observed feature-label pairs

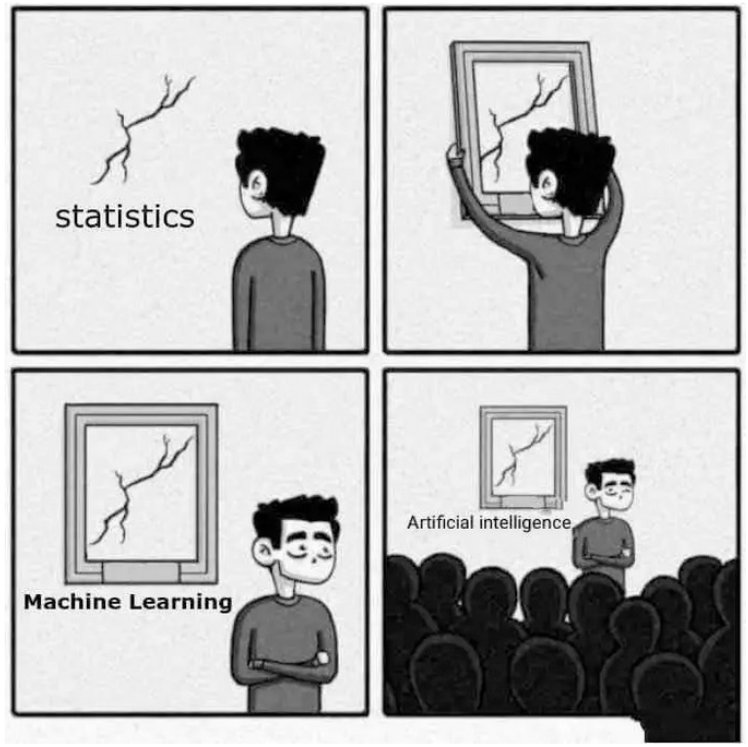
- Iteratively update beta to increase the model's likelihood, for each feature-label pair $(\boldsymbol{x}_i, y_i)$:

  - $\hat{\beta}_t = \hat{\beta}_{t-1} + \eta \frac{\partial}{\partial \beta} \log p(y_i | x_i; \beta)$

  - η: learning rate, controls how big each update is

Recap: machine learning

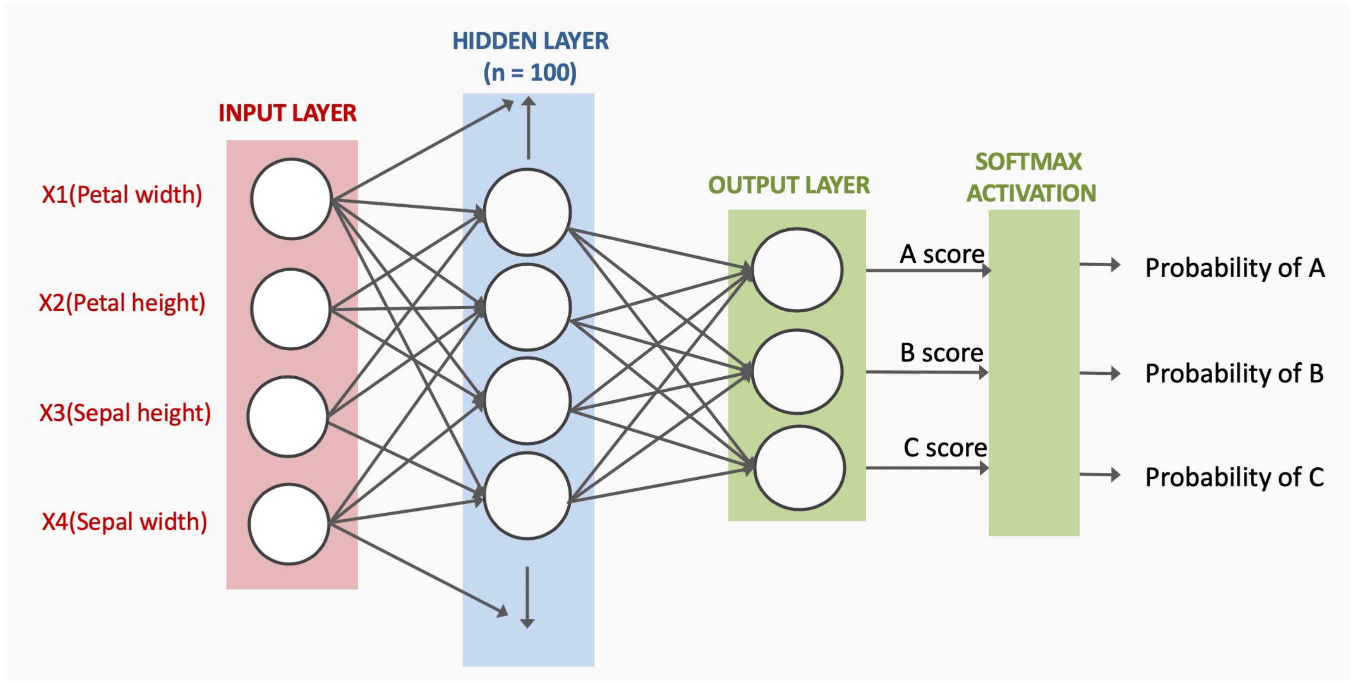
- Statistics vs machine learning vs deep learning (aka artificial intelligence)
- They are closely connected, but have developed important, specialized subfields



original comic by sandserif

Deep Neural Networks: Multi-layer perceptron (MLP)

- In its most basic form, DNNs are stacks of logistic regression.

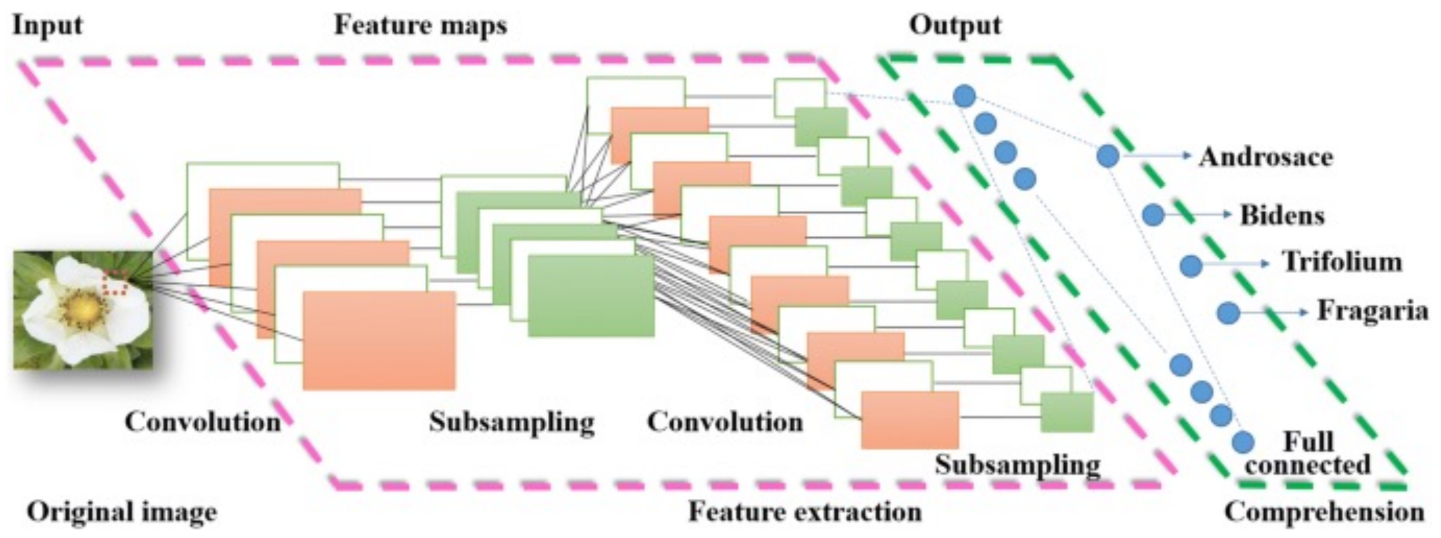


Each one neuron of n=100 in the hidden layer is a logistic regression!

https://harvard-iacs.github.io/2020-CS109A/lectures/lecture30/notebook-2/

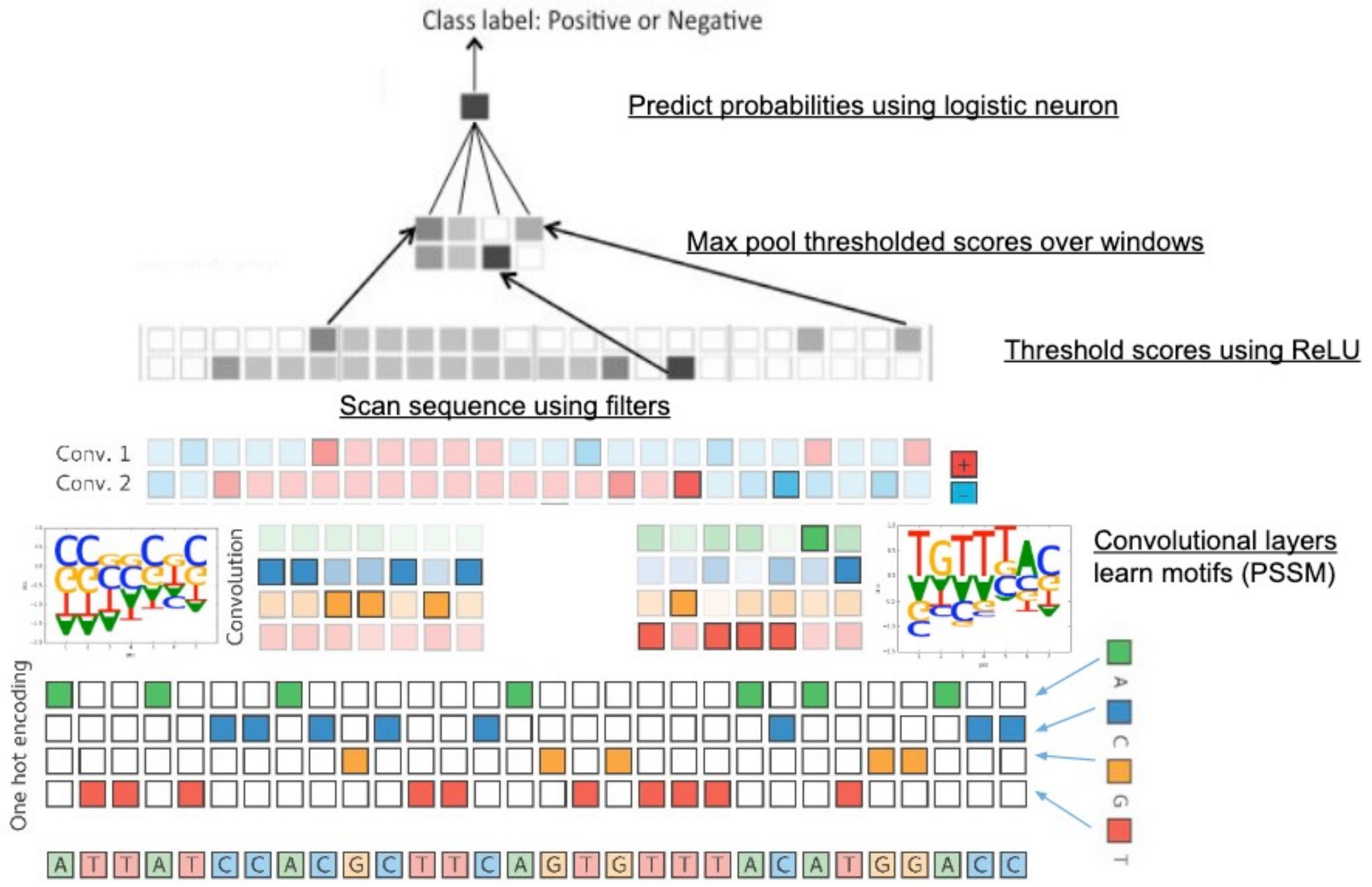
Convolutional Neural Networks on Images

- CNNs are special DNNs that are good at learning local spatial patterns.
- These patterns eliminate the need for manual features (e.g. petal length).
- However, Exact interpretation of image patterns is an active research direction.
    - In contrast, CNN interpretations on genomics are more straightforward.



Deep convolutional neural network for automatic discrimination between *Fragaria* × *Ananassa* flowers and other similar white wild flowers in fields. Plant Methods, 2018.

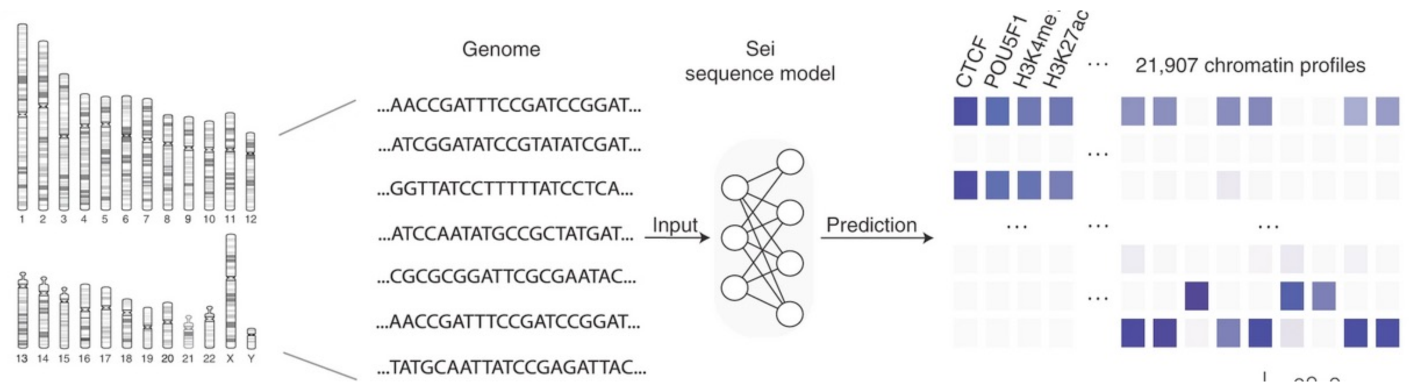
Convolutional Neural Networks on Genomics


Class label: Positive or Negative
Predict probabilities using logistic neuron
Max pool thresholded scores over windows
Threshold scores using ReLU
Scan sequence using filters
Conv. 1
Conv. 2
+
-
Convolution
Convolutional layers learn motifs (PSSM)
One hot encoding
A
C
G
T
A T T A T C C A C G C T T C A G T G T T T A C A T G G A C C

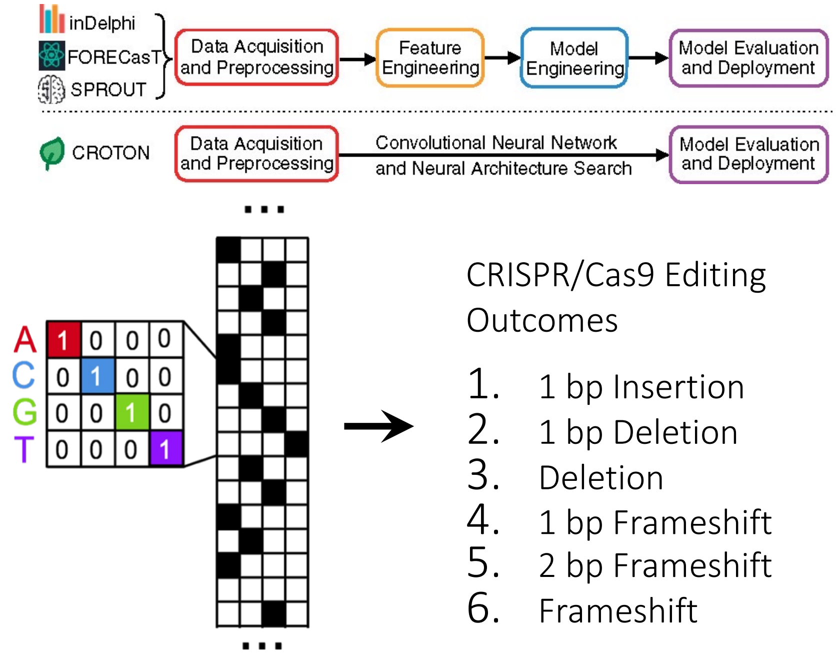
How to train your DragoNN.
https://www.genome.gov/sites/default/files/Multimedia/Slides/ENCODE2016-ResearchAppsUsers/Anshul_slides.pdf

Effective models of various molecular variations

- CNNs were first developed to predict epigenetic markers from DNA sequences
- Later introduced to vast different molecular variations, such as enhancers, gene expression, RNA splicing/polyadenylation.


Genome
Sei sequence model
...AACCGATTTCCGATCCGGAT...
...ATCGGATATCCGTATATCGAT...
...GGTTATCCTTTTTATCCTCA...
...ATCCAATATGCCGCTATGAT...
...CGCGCGGATTCGCGAATAC...
...AACCGATTTCCGATCCGGAT...
...TATGCAATTATCCGAGATTAC...
Input
Prediction
CTCF
POU5F1
H3K4me
H3K27ac
21,907 chromatin profiles

Sei
Chen et al., Nature Genetics, 2022


inDelphi
FORECasT
SPROUT
Data Acquisition and Preprocessing
Feature Engineering
Model Engineering
Model Evaluation and Deployment
CROTON
Data Acquisition and Preprocessing
Convolutional Neural Network and Neural Architecture Search
Model Evaluation and Deployment
A
C
G
T

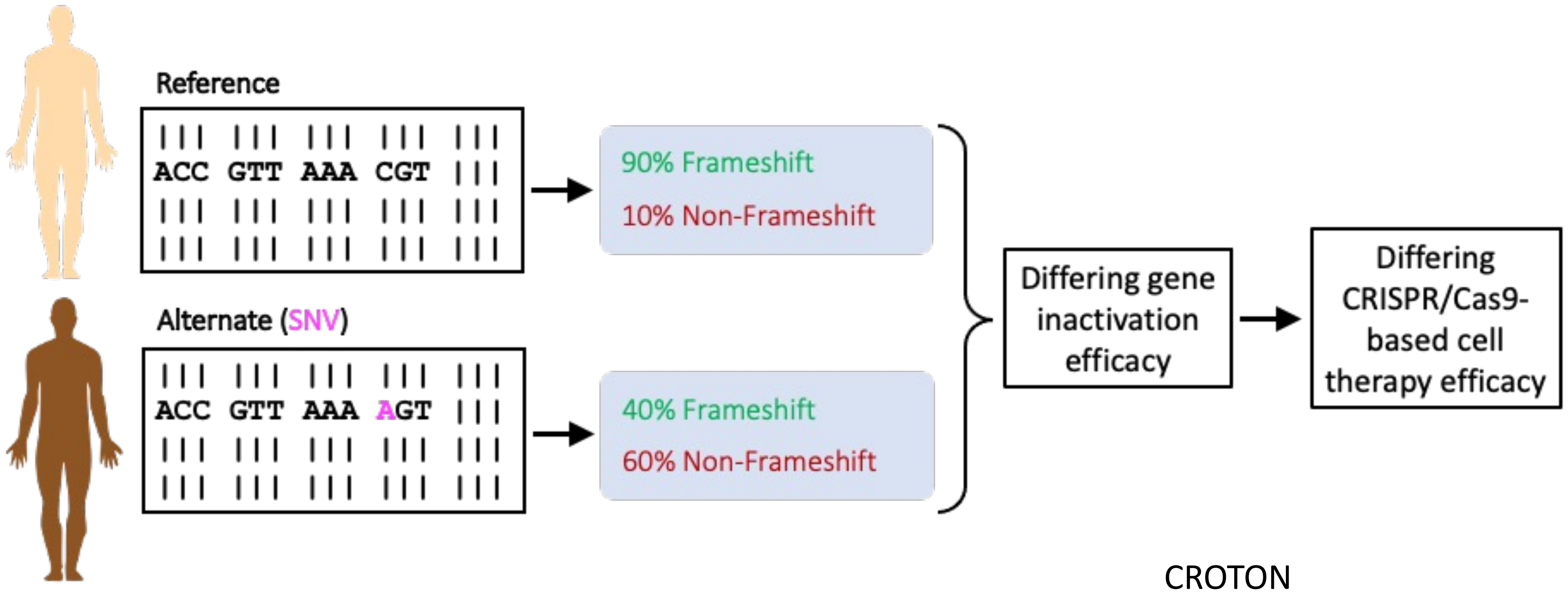
CRISPR/Cas9 Editing Outcomes

1. 1 bp Insertion
2. 1 bp Deletion
3. Deletion
4. 1 bp Frameshift
5. 2 bp Frameshift
6. Frameshift

CROTON
Li et al., Bioinformatics, 2022

Interpretations of functional genetic variations

- Central dogma established the causality direction.
- With a CNN model, we can introduce mutations *in silico* and predict its molecular effect.



CROTON
Li et al., Bioinformatics, 2022

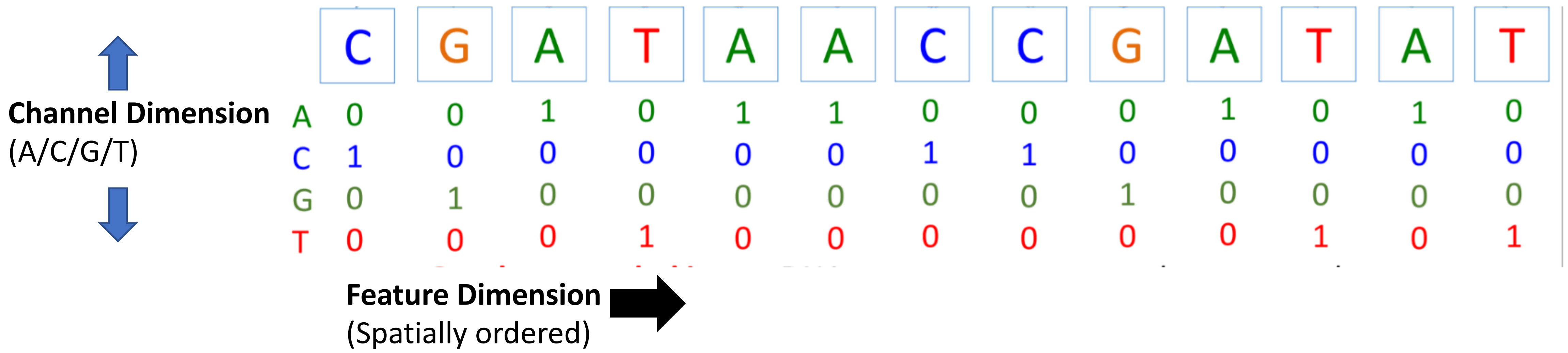
CNNs provide useful tools in genomics

- In summary, CNNs are powerful tools in genomics for these reasons:
  - end-to-end modeling without manual feature engineering
  - flexibility to model multiple relevant tasks
  - ability to introduce *in silico* mutations

- Next, we will study and implement a CNN in a step-by-step guide.

Genomic Sequence Input

- Convert DNA sequences to one-hot encoded matrix



	C	G	A	T	A	A	C	C	G	A	T	A	T
A	0	0	1	0	1	1	0	0	0	1	0	1	0
C	1	0	0	0	0	0	1	1	0	0	0	0	0
G	0	1	0	0	0	0	0	0	1	0	0	0	0
T	0	0	0	1	0	0	0	0	0	0	1	0	1

Train-Validation-Test Split

- We held-out a random portion of our labelled data to evaluate how well the model will generalize to new cases.
- For many ML methods and DL in particular, an additional validation dataset is used to assess overfitting during training and for tuning hyperparameters.
- Be cautious about potential data leakage-
  - In real human data, it is best practice to hold-out some chromosomes.

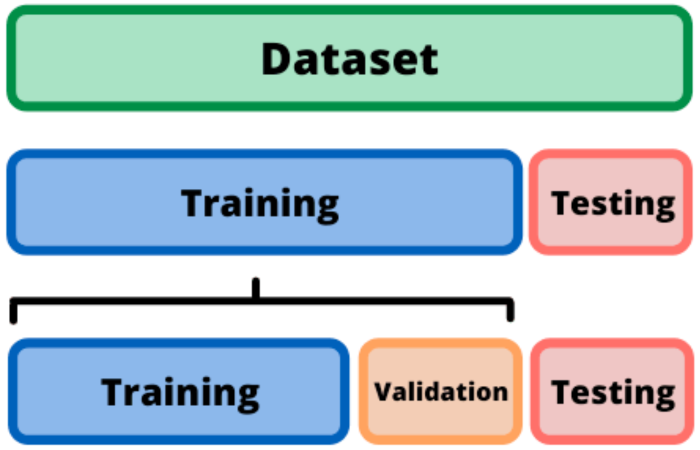
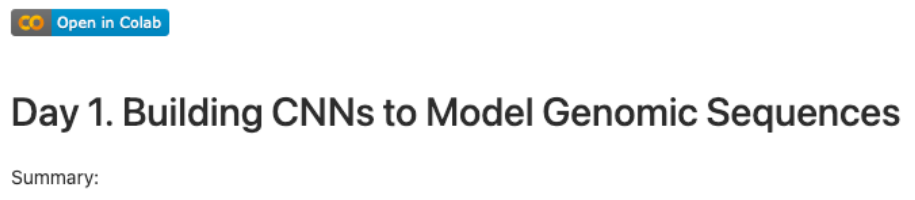


Illustration of dataset split.
Source.

Practice 1: Prepare Datasets

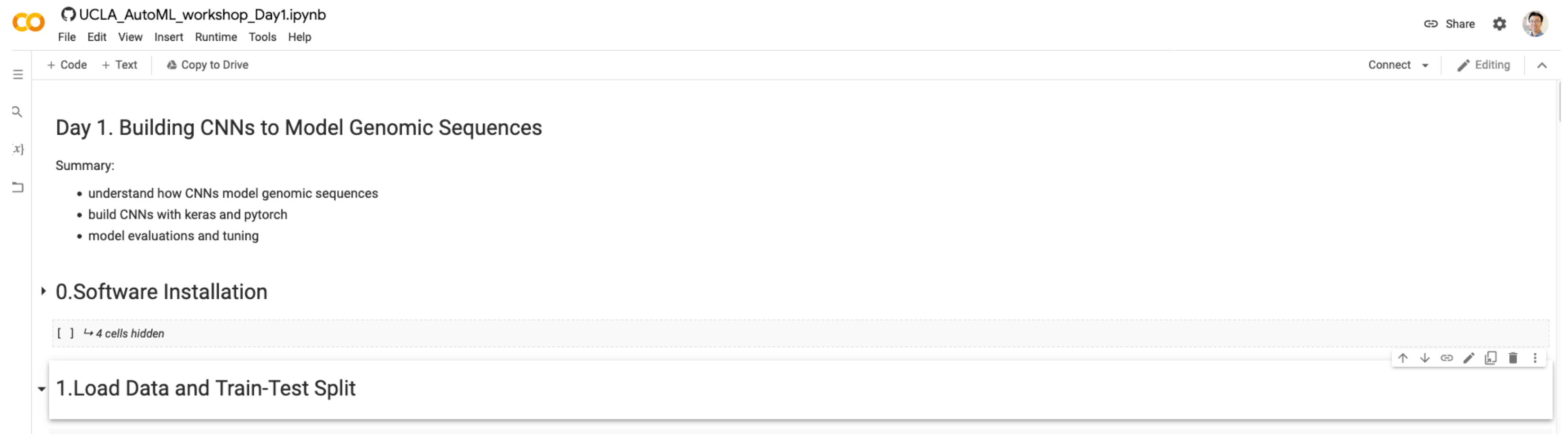
- Go to Github Repo and find Day 1 Jupyter notebook
- Click on "Open in Colab"



Day 1. Building CNNs to Model Genomic Sequences

Summary:

- In Google Colab, connect to a new Runtime
- Run from the beginning till the end of Section 1.Load Data and Train-Test Split



Day 1. Building CNNs to Model Genomic Sequences

Summary:

- understand how CNNs model genomic sequences
- build CNNs with keras and pytorch
- model evaluations and tuning

0.Software Installation

[ ] ↳ 4 cells hidden

1.Load Data and Train-Test Split

Practice 1: Prepare Datasets

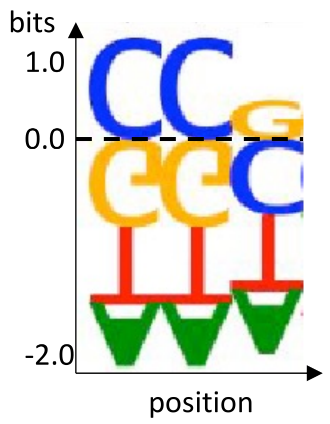
- 1. Exploratory data analysis on the positive and negative label proportions in training/validation/test sets.

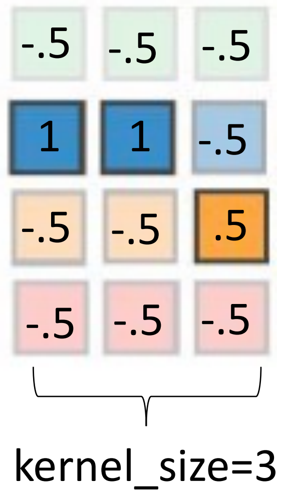
- 2. Exploratory data analysis on the one-hot encoded feature matrices – proportions of A/C/G/T.

- 3. Write your own Python function to convert one-hot matrix back to DNA sequence, and the inverse function that convert DNA sequence to one-hot matrix.

Convolution

- A convolutional kernel is a matrix that can be translated to a motif positional weight matrix.
- The weights are learned through gradient descent (similar to logistic regression's betas).



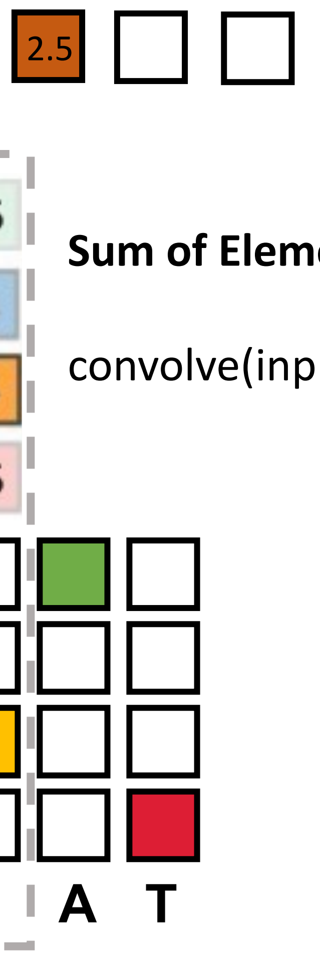


-.5	-.5	-.5
1	1	-.5
-.5	-.5	.5
-.5	-.5	-.5

kernel_size=3

Convolution

- Convolution operation applies a kernel matrix to scan the input sequence.
- Output dim = input feature dim – kernel size + 1 = 5 – 3 + 1 = 3



Convolution kernel

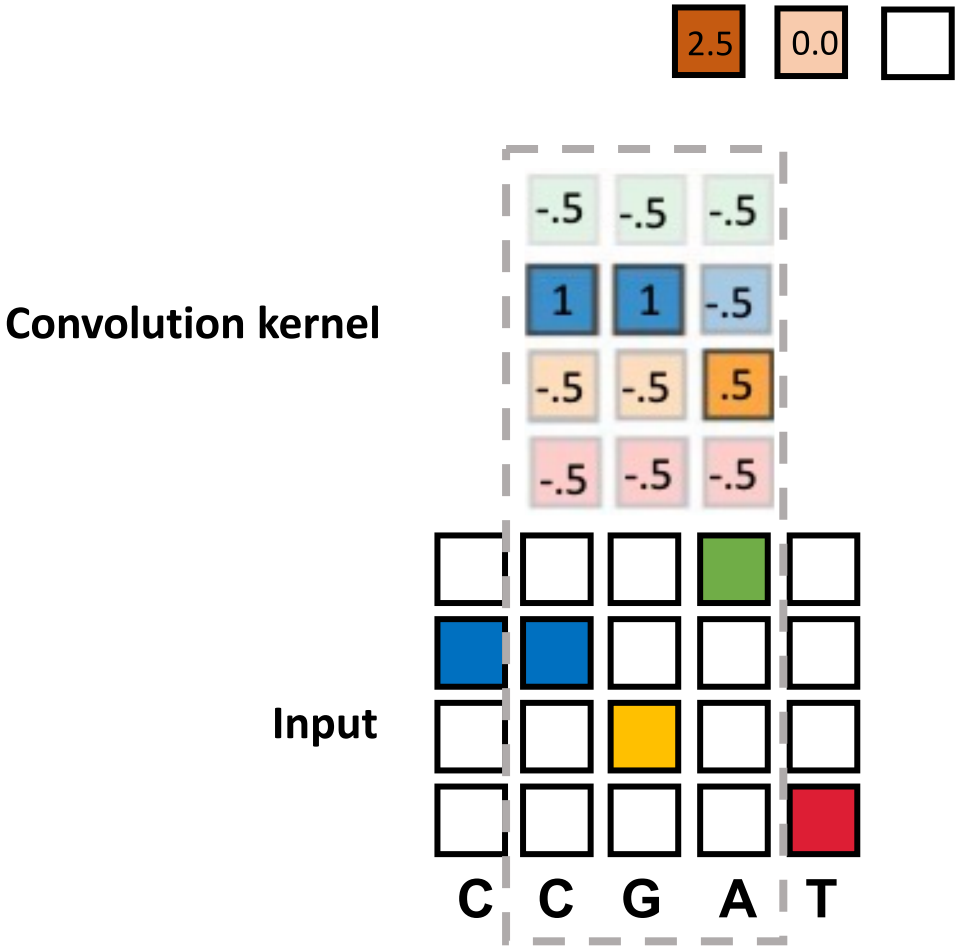
Input

Sum of Element-wise multiplication

convolve(input[0:3], kernel) = 2.5

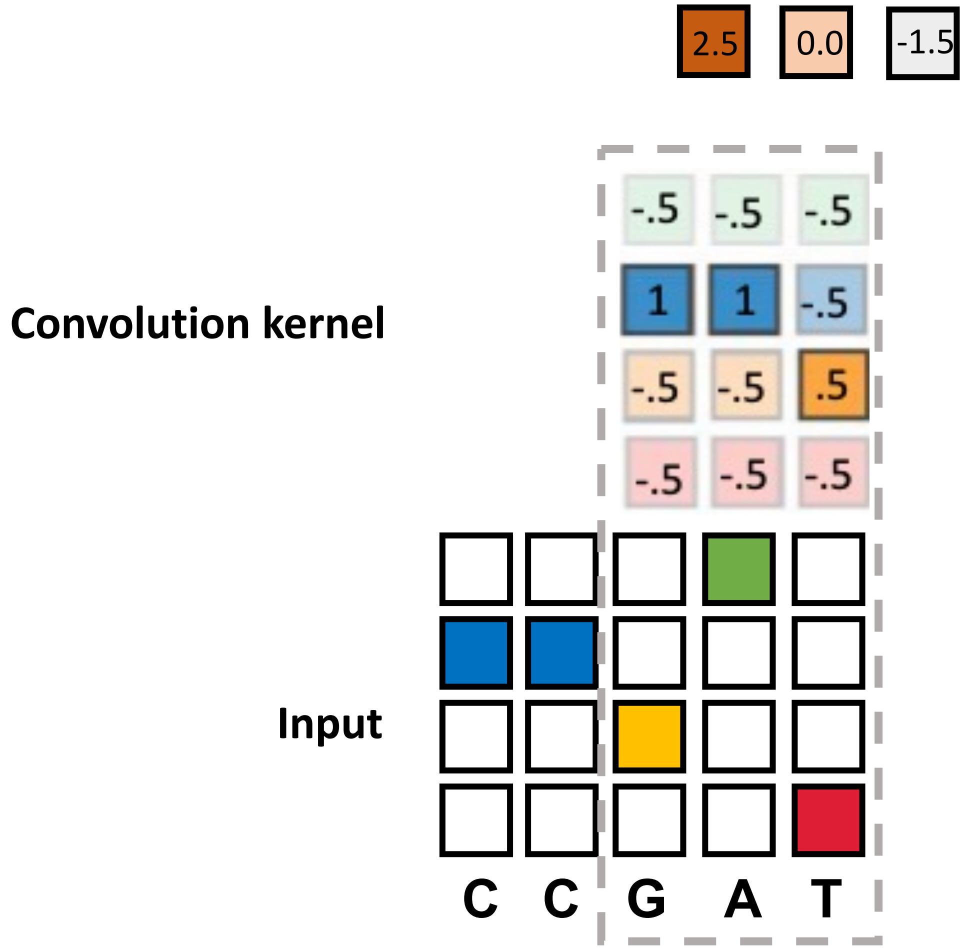
Convolution

- Convolution operation applies a kernel matrix to scan the input sequence.
- Output dim = input feature dim – kernel size + 1 = 5 – 3 + 1 = 3



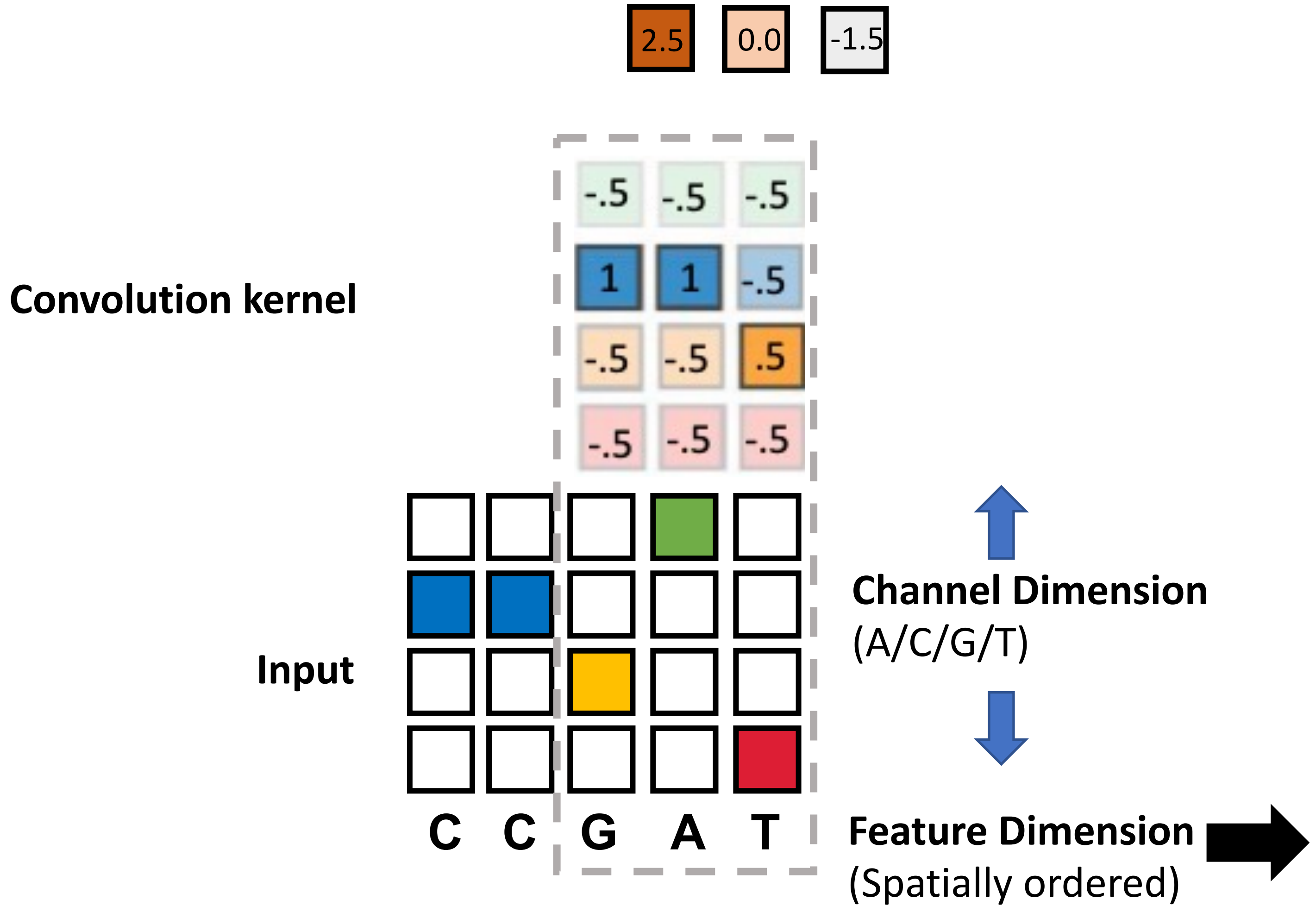
Convolution

- Convolution operation applies a kernel matrix to scan the input sequence.
- Output dim = input feature dim – kernel size + 1 = 5 – 3 + 1 = 3



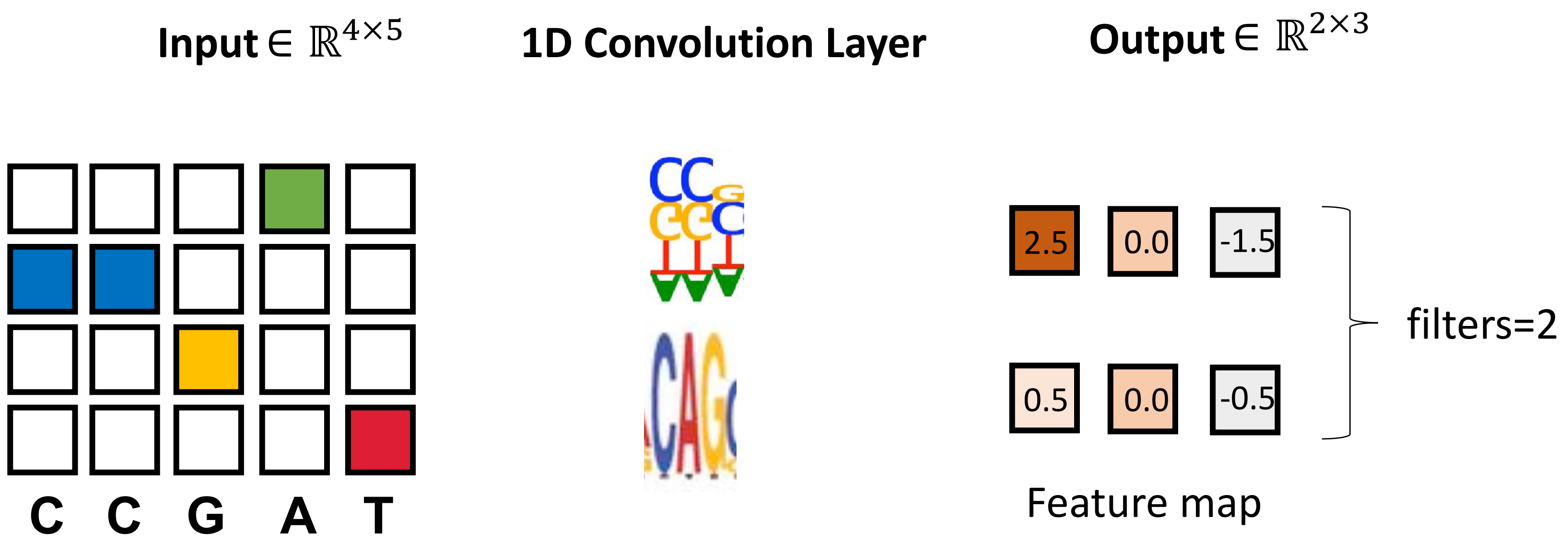
Convolution

- Convolution operation applies a kernel matrix to scan the input sequence.
- Output dim = input feature dim – kernel size + 1 = 5 – 3 + 1 = 3



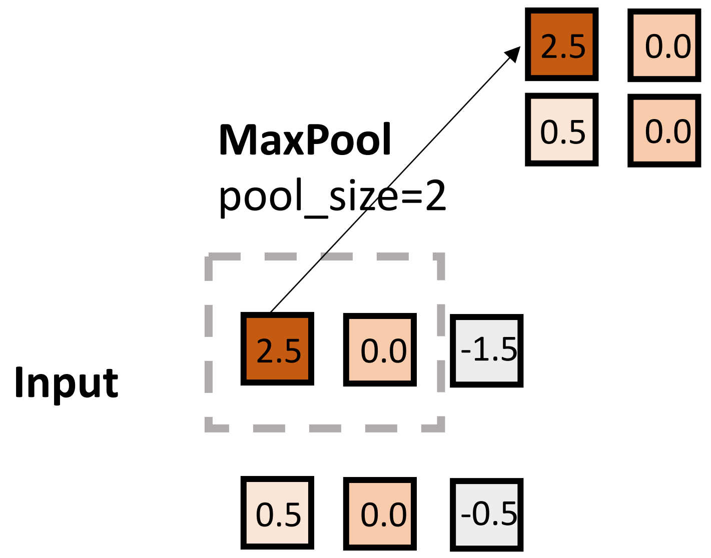
Convolution

- A convolution layer is a collection of convolution kernels.
  - Input: feature dim=5 (5bp), channel dim=4 (A/C/G/T)
  - Conv: filters=2, kernel size=3
  - Output: feature dim=3, channel dim=2



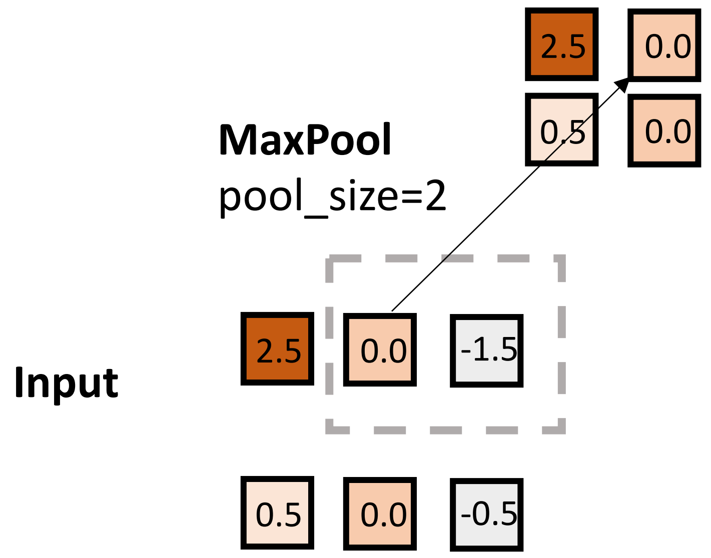
Pooling

- Scan the input feature map with a sliding window on the feature dimension, and keep the max or average values within each window.
- Pooling changes the feature dim, but not channel dim.



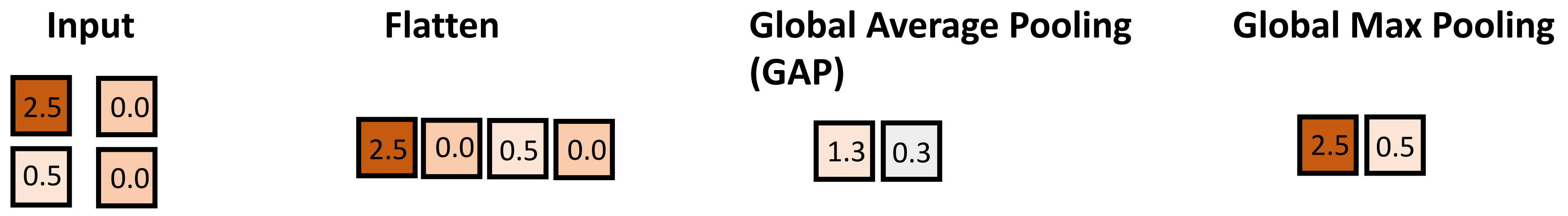
Pooling

- Scan the input feature map with a sliding window on the feature dimension, and keep the max or average values within each window.
- Pooling changes the feature dim, but not channel dim.



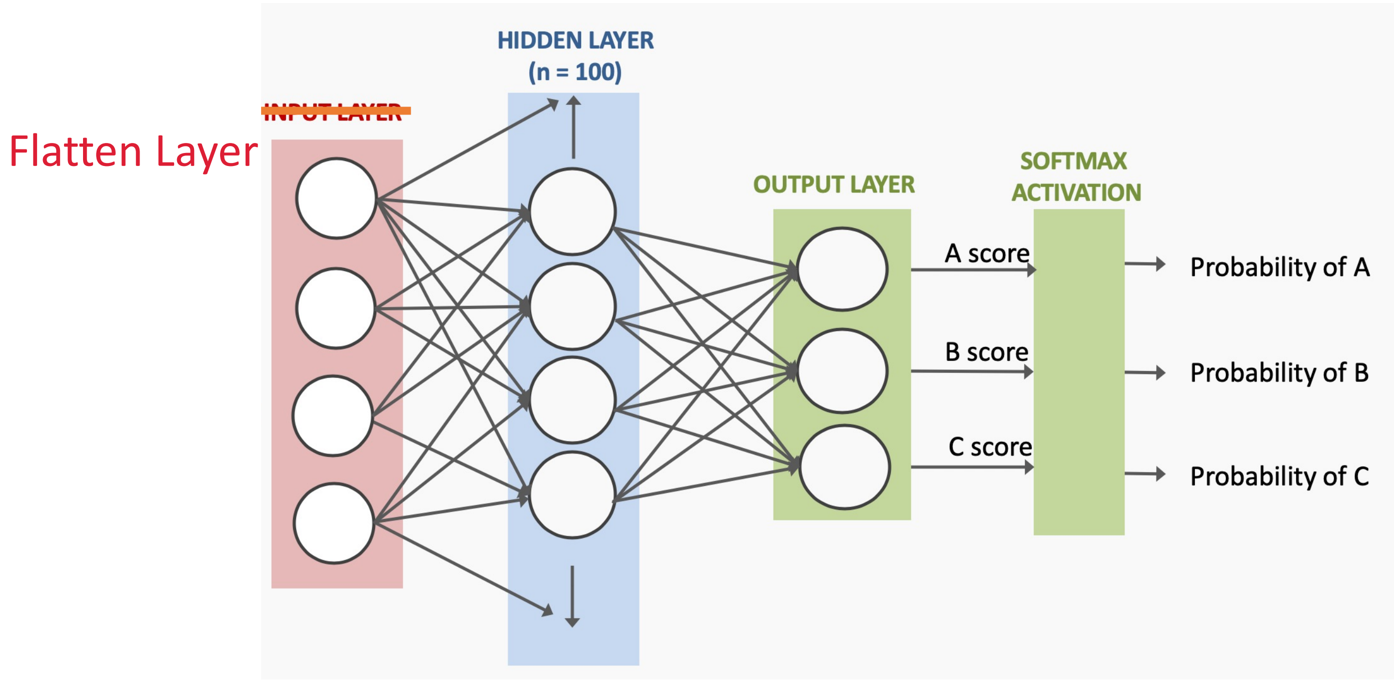
Flatten

- Flatten operations will reshape the feature map from a matrix to a vector.



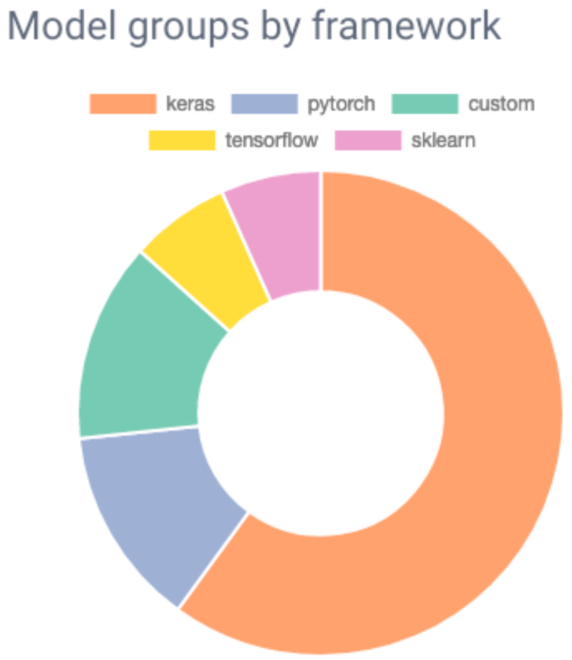
Fully Connected

- The previous flatten layer brings us back to the multi-layer perceptron.



Why Keras?

- Keras is excellent for fast prototyping and research development.
- The Model API implemented many useful callbacks and metrics with a self-contained design.
- Keras is flexible for advanced models through custom layers.



Kipoi: Model zoo for genomics

https://kipoi.org/

Practice 2: Build CNN in Keras

Run Section 2.Build a CNN Model using Tensorflow/Keras Open in Colab

- Change the parameter values of filters, kernel_size, pool_size, and observe the shape changes in model.summary().
- Change the flatten layer from GAP to Flatten.
- Uncomment Tensorboard to visualize training results.

Understanding Keras Model API

- model.compile(): configure optimizations

- *optimizer:* gradient descent and its variants; Adam is usually a good default choice.
- *loss:* loss function; must be differentiable to parameters.
  - classification: binary_crossentropy, categorical_crossentropy
  - regression: mse, mae
- *metrics:* additional monitors for model performance, such as AUROC (will cover next).

Understanding Keras Model API

- model.fit(): train model parameters

- *batch_size:* compute gradients on this number of samples
- *epochs:* how many epochs to train; a complete iteration through all train data is one epoch.
- *callbacks:* a list of callback functions that will be triggered upon epoch end.
  - TensorBoard: log model performance to tensorboard
  - ModelCheckpoint: save model parameters
  - EarlyStopping: stop training earlier than specified epochs, if model starts overfitting

Comparing PyTorch vs Keras

- PyTorch is another popular deep learning framework.
- Excellent for research and development of novel models, with more exposure and control over the model’s behavior.

- Run Section 3.Build a CNN using Pytorch/Lightning. CO Open in Colab

Building CNNs in PyTorch

- In general, PyTorch model has a slightly higher requirement Python Programming:
    - Models are subclassing torch.nn.Module
    - Operators and Tensors are built during __init__()
    - Output channel dims are determined by users
    - model.forward() sets the forward pass
    - train loops, metrics are custom-built, or relies on other packages (e.g. lightning, torchmetrics)

<u>Can you match the PyTorch code implementations to Keras syntax?</u>

Model Evaluations

- Evaluations are essential for model engineering and its final deployment.

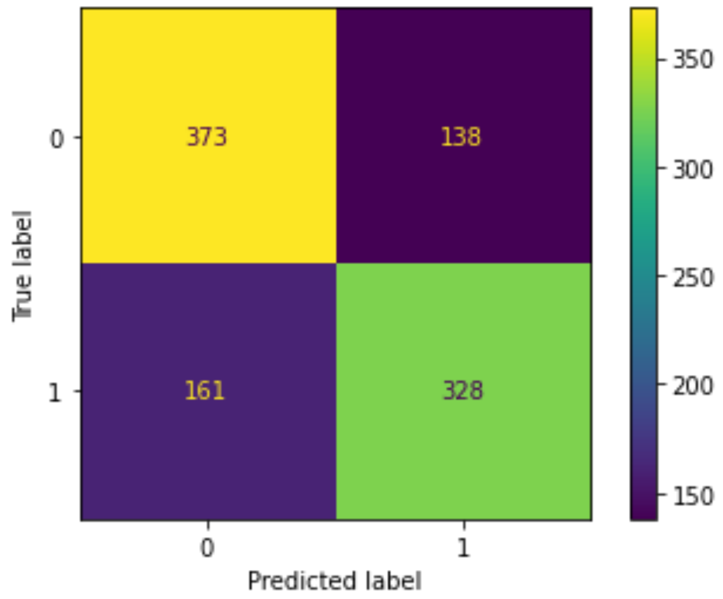
- For classification tasks, there are two types of evaluation metrics:
  - Discrete: use a threshold to binarize the predicted scores to 0/1. Compute the concordance between predicted labels and observed labels.

  - Continuous: keep predicted scores as continuous values. Compute the prediction consistence with observed labels over multiple thresholds.

See Also- Day 2 from the workshop: Machine Learning with Python

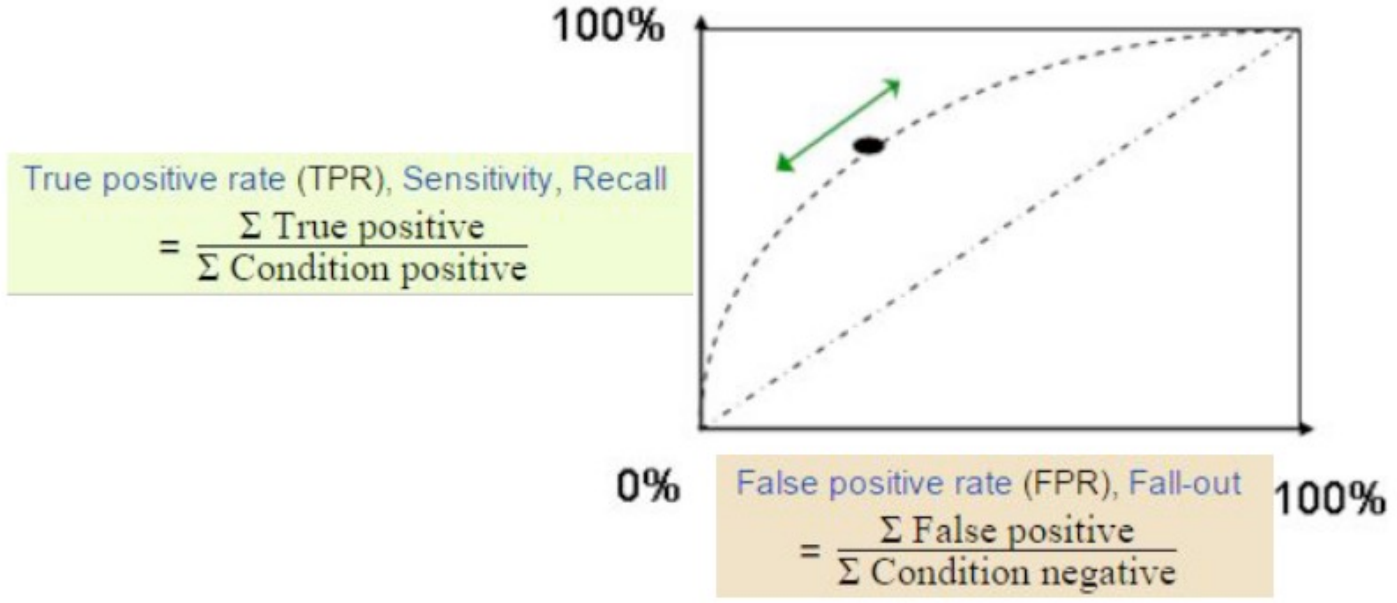
Discrete prediction evaluations

- Definition of TP, FP, TN, FN, f1 score.
- Confusion matrix is useful for visualizations, especially for multi-class predictions.



Continuous prediction evaluations

- Area under the Receiver operating curve (AUROC)
- Compares sensitivity (recall) to false positive rate (1-specificity) at various thresholds.
  - auROC = 1 (Perfect classifier)
  - auROC = 0.5 (Random classifier)

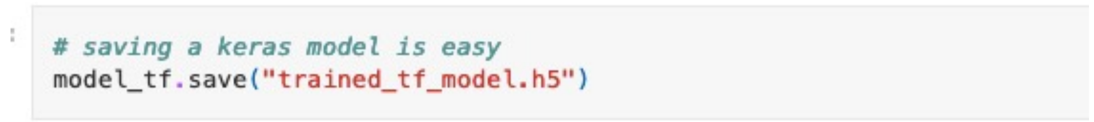

100%
True positive rate (TPR), Sensitivity, Recall
$= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$
0%
False positive rate (FPR), Fall-out
$= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$
100%

How to train your DragoNN.
https://www.genome.gov/sites/default/files/Multimedia/Slides/ENCODE2016-ResearchAppsUsers/Anshul_slides.pdf

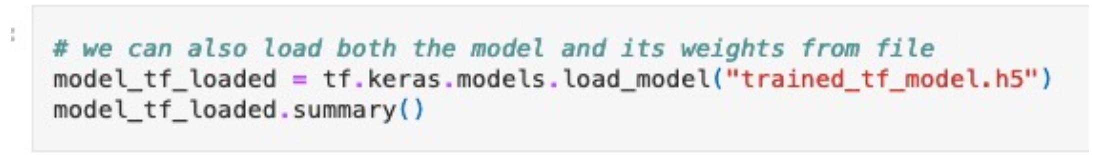
Model Save & Load

- Now that we have a preference over multiple models, it’s essential to save a trained model and reload it for inference.

Keras

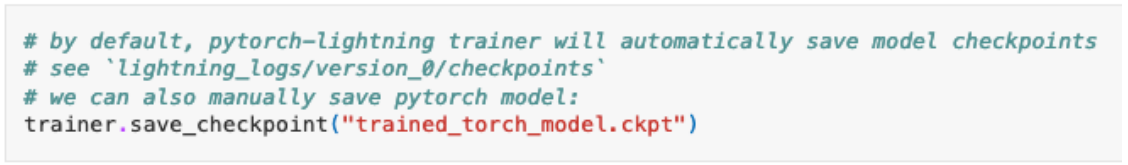


```
# saving a keras model is easy
model_tf.save("trained_tf_model.h5")
```

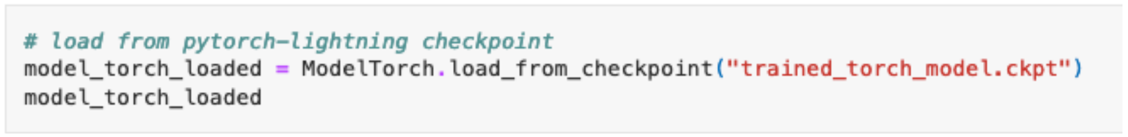


```
# we can also load both the model and its weights from file
model_tf_loaded = tf.keras.models.load_model("trained_tf_model.h5")
model_tf_loaded.summary()
```

PyTorch



```
# by default, pytorch-lightning trainer will automatically save model checkpoints
# see `lightning_logs/version_0/checkpoints`
# we can also manually save pytorch model:
trainer.save_checkpoint("trained_torch_model.ckpt")
```



```
# load from pytorch-lightning checkpoint
model_torch_loaded = ModelTorch.load_from_checkpoint("trained_torch_model.ckpt")
model_torch_loaded
```

Practice 3: Model Evaluations

Run Section 3-5.Build a CNN Model using PyTorch/Lightning, Model Inference and Evaluation and Model Saving and Loading

Open in Colab

- Understand PyTorch CNN model.

- Improve Model performance as evaluated by the metrics we covered.

- Compare different model's performance using different evaluation metrics, model architectures, and optimization configurations.

Summary & What’s Next

- Understand how CNNs model genomic sequences
- Build CNNs with Tensorflow and PyTorch
- Model evaluations and tuning

- Tomorrow:
  - Framework-agnostic CNN models
  - Automatically find accurate CNNs in under 5min

Q&A

Write your questions in this Google Doc.